

FIG. 1

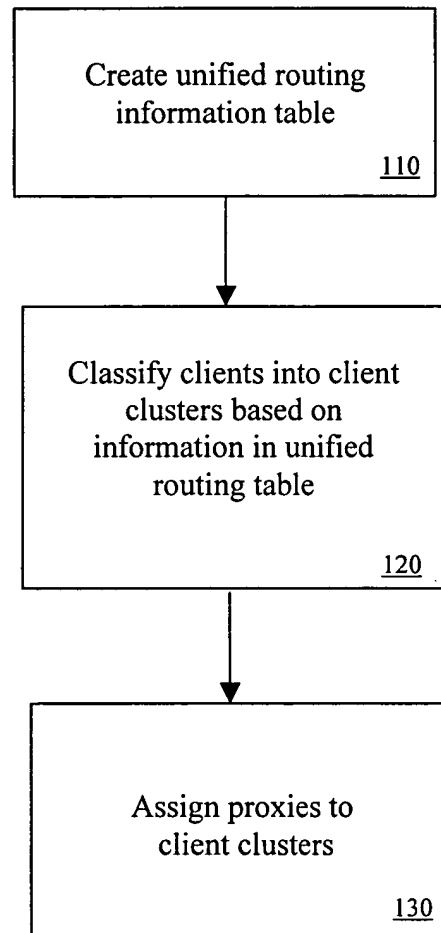


FIG. 2

190

FIG. 3

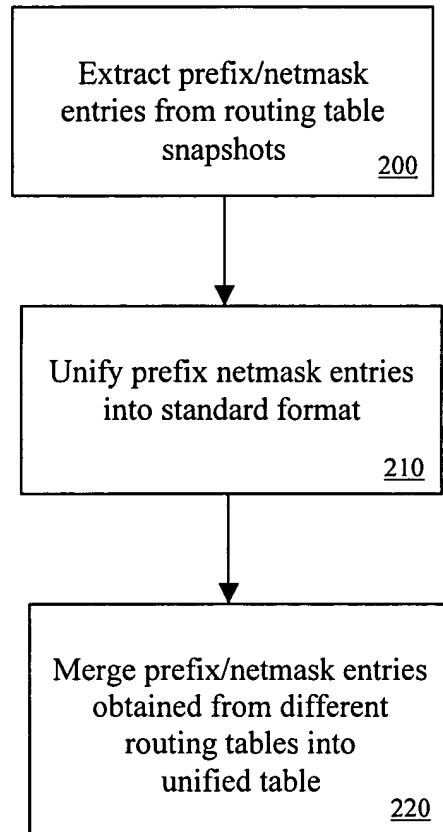


FIG. 4

	230		240	250
Formats	x1.x2.x3.x4/k.1k.2k.3k.4		x1.x2.x3.x4/l	x1.x2.x3.0
Routing Tables	MAE-EAST MAE-WEST PACBELL PAIX		ARIN AT&T CANET NLANR VBNS	CANET
Examples	193.1/255.255 193.0.128/255.255.192		128.148.0.0/16	130.15.0.0
Unification	193.1/255.255 193.0.128/255.255.192		128.148.0.0/255.255	130.15.0.0/255.255 192.75.72/255.255.255

FIG. 5

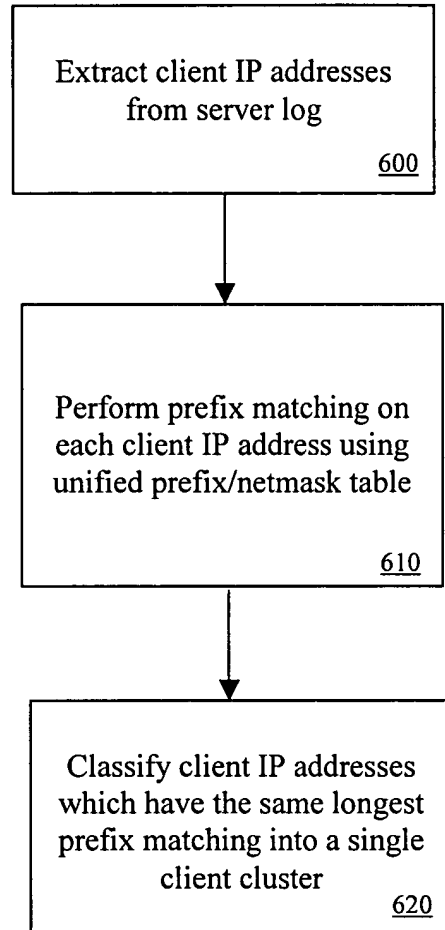


FIG. 6



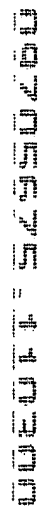


FIG. 8



matching code for the <S,R,H> values generated by the first part of the algorithm

```

5  /*
   * multi-level retrieve longest prefix match
   *
   * input arguments:
   *
   *   S      top level shift
10  *   R      retrieve internal nodes
   *   H      retrieve leaf nodes (next hop index)
   *   addr   IP address to match
   *
   * return:
15  *   0      no match
   *   >0     next hop index
   */

20 int
lpmatch(uint8 S, uint32* R, uint8* H, uint32 addr)
{
25     uint32      b;
     uint32      x;

     x = R[addr >> S];
     while (b = x >> 27)
     {
30         b = (x & ((1<<26)-1)) + ((addr>>(S-=b)) & (1<<b)-1));
         if (x & (1<<26))
         {
             x = H[b];
             break;
         }
35         x = R[b];
     }
     return x;
}

```

FIG. 9a

```

/*
 * 2-level retrie longest prefix match
 *
 * input arguments:
 *
 *   S      top level shift
 *   R      retrie internal nodes
 *   H      retrie leaf nodes (next hop index)
 *   addr   IP address to match
 *
 * return:
 *
 *   0      no match
 *   >0     next hop index
 */

```

```

int
lpmatch2(uint8 S, uint32* R, uint8* H, uint32 addr)
{
    uint32      x;

    x = R[addr >> S];
    return H[(x & ((1<<26)-1)) + ((addr & ((1<<S)-1)) >> (S - (x>>27)))];
}

```

FIG. 9b